Journal of Intelligent & Robotic Systems
https://doi.org/10.1007/s10846-017-0757-5

Quadrotor Autonomous Approaching and Landing on a Vessel Deck

Liyang Wang[1] · **Xiaoli Bai**[1]

Received: 2 August 2017 / Accepted: 3 December 2017

Abstract
Autonomous landing of a quadrotor UAV on a vessel deck is challenging due to the special sea environment. In this paper, we present an on-board monocular vision based solution that provides a quadrotor with the capability to autonomously track and land on a vessel deck platform with simulated high sea state conditions. The whole landing process includes two stages: approaching from a long range and landing after hovering above the landing platform. Only on-board sensors are used in both stages, without external information input. We use Parrot AR.Drone as the experimental quadrotor platform, and a self-designed vessel deck emulator is constructed to evaluate the effectiveness of the proposed vessel deck landing solution. Experimental results demonstrate the accuracy and robustness of the developed landing algorithms.

Keywords Autonomous landing · Quadrotor · Vessel deck · Visual tracking

1 Introduction

Nowadays, unmanned aerial vehicles (UAVs) are playing critical roles in many areas [1]. Numerous applications exist that require autonomous maritime deployment and retrieval of a UAV, such as iceberg monitoring, coastal surveillance, and wildlife monitoring [2]. The vertical takeoff and landing capability (VTOL) makes the quadrotor particularly suitable for maritime applications, where the runway for takeoff and landing is often very short. However, autonomous landing of quadrotors on a ship deck is a challenging task owing to limited payload for sensors [3], the unreliable GPS measurements [2], and especially ocean waves' disturbance [4].

Without using GPS measurements, for the moving platform autonomous landing problem, vision based solutions are widely used, and various approaches have been studied [5]. Compared with the on-ground vision landing system, such as [6] and [7], on-board vision landing system is typically adopted, especially for landing on a moving platform [8]. For onboard vision solutions, some researchers are interested in stereo vision systems, since stereo vision systems are convenient to acquire depth information [9]. Reuben et al. present a stereo based method for the interception of a static or moving target [10]. Target detection and relative position estimation are realized by using two fisheye cameras. Because stereo based method needs two cameras, which leads to more weight on-board and increased cost, many researchers propose monocular vision based approaches. For example, Xudong et al. describe a UAV implemented with vision and laser based localization algorithm to track and land on a moving platform [11]. In that work, a LiDAR scanning range finder is used to determine the altitude. In [12], Benini et al. present a real-time GPU-based pose estimation system for UAV autonomously takeoff and landing. The monocular video stream is processed on-board, and the test results show that the proposed system is able to provide precise pose estimation with a frame rate of at least 30 fps and an image resolution of 640×480 pixels. About the camera type selection, some researchers use ordinary cameras. In [13], Francesco et al. committed to the ability of a vision algorithm to detect and track a given landing sign. Optimized adaptive thresholding technique is developed to ensure the robustness of the system in different illumination situations. Benini et al. present

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10846-017-0757-5) contains supplementary material, which is available to authorized users.

✉ Xiaoli Bai
xiaoli.bai@rutgers.edu

Liyang Wang
liyang.wang@rutgers.edu

[1] Department of Mechanical and Aerospace Engineering, Rutgers University, Piscataway, NJ 08854, USA

Published online: 26 December 2017

an indoor mini-UAV localization system in [14]. With on-board monocular vision and wireless sensor network, 10 cm localization accuracy has been achieved. Some researchers use infrared (IR) camera, which is able to work at dark environments [15]. For example, Karl et al. use infrared (IR) camera to track a pattern of IR lights in conditions without direct sunlight [16].

Compared with other moving landing platforms, a vessel deck environment is even more challenging. It may have significant pose disturbances due to the sea waves and the wind. Most studies have not demonstrated their approaches can work when using vessel deck emulators, which include both pose and heave motion. In [17], Botao et al. have presented a control structure that achieves the fast and accurate landing of quadrotor onto a vertically sinusoidally oscillating platform. Their control structure consists of three modules: a motion estimation module, a trajectory generation module, and a tracking control module. However, the roll and pitch movements of the platform are not included. In contrast to the flat landing platform, John et al. present a Laser-based guidance of a quadrotor for precise landing on an inclined surface [18]. However, the pose angle of the landing platform is fixed in the experiments. PITM et al. present a landing algorithm that can land a quadrotor on a pose oscillating platform in [19]. However, in their experiments, the quadrotor is suspended from a string to restrict it from drifting away, which is not realistic in real landing. Furthermore, the movement of the platform is achieved by two people using their hands. The dynamics of the landing platform is not a good emulation of a vessel deck, and the quadrotor is too close to people and could be very dangerous. The dynamics of the ship-deck is introduced and emulated in [4]. The pose of the ship deck with respect to the vehicle is estimated by a vision system. However, that work didn't include UAV state estimation and motion control methods. A more complete and realistic solution is presented in [20]. The authors provide a complete approach which lands a UAV on a kayak positioned 20 m from the shore. However, the indoor test did not take the pose change of the landing platform into consideration, and for the landing on the kayak, the water waves are 'mild' according to their description. More significantly, according to the experiment results, under the indoor test environment without any disturbance, the average landing accuracy is $\pm$ 12 cm from the center of the board, and the landing success rate is less than 90%, which is far away being satisfied for critical landing missions.

This paper presents a monocular vision based system consisting of an autonomous long range approaching method and an accurate landing algorithm using only on-board cameras and other sensors provided by AR.Drone. The main contributions of this work include two aspects. First, simple but effective long range target detection and approach algorithms are developed. Second, we propose a novel system architecture that addresses realistic vessel deck landing using only low-cost on-board sensors. The designed landing process includes two stages: approaching from long range and landing after hovering above the landing platform. For the first stage, a robust method is designed to drive the done to an area above the landing platform. In this stage, the red color LEDs are used for landing platform tracking. The field of view (FOV) orientation of the front camera of the AR.Drone is redesigned to help searching the landing platform, and the state estimation is based on the new geometric relationship between the camera and the landing platform. We use a PID controller to stabilize the heading angle towards the landing platform. For the second stage, the drone lands autonomously on the landing platform accurately. We accomplish the landing sign recognition using computer vision solutions, and 'H' landing sign will be tracked once it appears in the FOV of the bottom camera. The state estimations are performed using on-board low-cost IMU, ultrasonic altimeter, and the image informations. Kalman filters are used to further improve the estimation accuracy. Lastly, we use PID controllers to execute the flight controls. Additionally, a vessel deck emulator, which can emulate movements of a vessel deck in high sea state conditions, is designed and used to validate the reliability and robustness of the proposed architecture. This emulator includes an 'H' landing sign surrounded by 4 red LEDs, and it provides a realistic emulation of the pose and heave motion of a vessel deck.

The organization of this paper is as follows. In Section 2, the background information of the components of the system is presented, including AR.Drone, the self-designed landing platform, and a Vicon Motion Capture System. Coordinate conventions are also introduced. Section 3 introduces how the landing platform is designed and how it works. The long range landing platform detection and tracking are presented in Section 4, including camera configurations, methods to recognize the landing sign, state estimation algorithms, and approaching procedure. Section 5 introduces the hovering and landing stage when the drone approaches to the landing platform. Image processing methods, state estimation algorithms, motion control, and the architecture of algorithms are presented. Indoor experiment results are presented in Section 6. Finally, conclusions are presented and opportunities for future work are discussed in Section 7.

2 Back Ground Information

Here we present the hardware and system architecture we developed for the testbed. Because the on-board

camera and sensors, the landing platform, and the motion capture system which is used as ground truth are working in different coordinate frames, we will also define the coordinate frames and introduce the transformation among them in this section.

2.1 Testbed Description

2.1.1 Parrot AR.Drone

The micro UAV we used in this work is a Parrot AR.Drone. AR.Drone is a quadrotor helicopter that has a 6 degree of freedom internal IMU, an ultrasonic altimeter, a forward-looking (front) camera and a downward-looking (bottom) camera [21], with a 640×360 resolution for both cameras. For the front camera, we measured the horizontal FOV is about 60°, and the vertical FOV is about 30°. For the bottom camera, we measured the horizontal FOV is about 40°, and the vertical FOV is about 25°.

With the sensors equipped on-board, altitude z is determined by a ultrasonic altimeter, roll angle ϕ and pitch angle θ are calculated by using data from an IMU. Estimations of these states are directly available from the Parrot AR.Drone. Although yaw angle ψ is also provided, we don't use it due to its low accuracy – the yaw drift is about 12° per minute when flying and about 4° per minute when in standby [22]. For yaw angle ψ and relevant position estimation, our solution is using vision algorithms.

According to [22], there are 4 control channels to control AR.Drone: (1)Roll angle, (2)Pitch angle, (3)Angular velocity of yaw, and (4)velocity of the altitude direction. We denote them as:

$$u_1 = \phi \tag{1}$$

$$u_2 = \theta \tag{2}$$

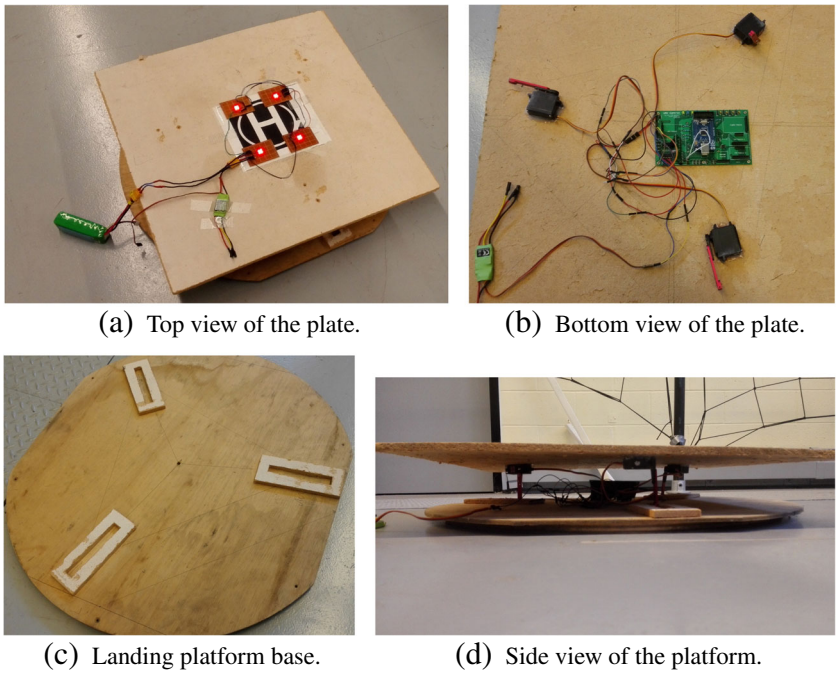
$$u_3 = \dot{\psi} \tag{3}$$

$$u_4 = \dot{z} \tag{4}$$

Because of the limited capability of the ARM processor equipped on AR.Drone [22], we run the system code on a ground computer and stream commands to the UAV over a Wi-Fi connection. The UAV streams IMU data and camera images to the ground computer also over the Wi-Fi connection. From our experiments, the delay of the Wi-Fi connection do not cause the failure of the state estimation as well as the control.

2.1.2 Vessel Deck Emulator

We emulate the movement of the ship deck in the sea, using our self-designed landing platform. See Fig. 1. The motivation to design this landing platform is to emulate pose change and heave motion of a vessel deck. The ship can be modelled as a rigid body moving in the sea [1, 4], such that the movement of a ship deck can be simulated by an attitude-programmable plate. The concept that 3 points define a plane is used in the design of the landing platform. The same idea is adopted in designing the landing platform in [23].



(a) Top view of the plate. (b) Bottom view of the plate.

(c) Landing platform base. (d) Side view of the platform.

Fig. 1 The landing platform

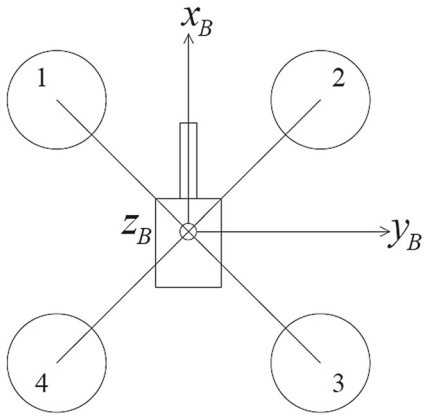


Fig. 2 Body frame

The landing platform includes 2 pieces: a square plate and a bottom base. The square plate is powered by 3 independent servos deployed on the back, and they are located at the 3 vertices of an equilateral triangle. Each servo is connected with a rigid arm at the joint, which is used to change the height of that vertex. The servos are controlled by a STM32 ARM processor, which sends PWM signals to change the angle of arms. On the top side of the square plate, an 'H' sign for landing is attached at the center. Besides that, 4 red LEDs are deployed at the 4 corners of the 'H' sign. These LEDs are used to guide the drone from long distances.

The function of the base is to hold the plate. There are 3 sliding grooves. We set the arms in those grooves and make sure the arms move in a limited area, so the plate will not move away from the base. The details of the landing platform is introduced in Section 3.

2.1.3 Vicon Motion Capture System for Ground Truth

We also use a Vicon Motion Capture System (including 8 Bonita cameras capture up to 250 fps with one megapixel of resolution) to provide state measurements for the quadrotor, which is assumed as the ground truth. We emphasize that, the measurements from Vicon are only used for validation, and the drone only uses the information from on-board sensors to realize its autonomous flight.

2.2 Coordinate Frames and Transformations

Four coordinate frames are used in this work. The first 3 coordinate frames are 3D right-handed, and the unit is meter. The inertial or east-north-up (ENU) frame, denoted as N, is the basis for Vicon measurements and provides the inertia position. The origin is set to be the center of our experimental site. East, north, and up will be referred to as the axis x_N, y_N, and z_N, respectively. The quadrotor body frame B, with the origin to be the bottom camera's lens' center, x_B axis between rotors 1 and 2, pointing to the optical axis of the front camera, y_B between rotors 2 and 3 and z_B pointing to the optical axis of the bottom camera, is used to define the vehicle motion. Because the rigid connection between the body of quadrotor and the bottom camera, which we use as the image source, the body frame B also works as the camera frame. Local reference frame L works as the intermediate frame of the inertial frame and the body frame. The origin of local reference frame is same as the body frame, while x_L points to the north, y_L points to the east, and z_L points downward. See Figs. 2 and 3.

The last frame is the image frame I, used for both the front camera and the bottom camera. Frame I is a 2D frame, with the origin O_I to be the top-left point of the image, U axis along the length, and V axis along the width. The unit of image frame is pixel. See Fig. 4. The coordinate of an arbitrary point P is (u, v) with respect to the image frame.

Figure 5 shows the optical geometric relationship between the bottom camera and the drone's body. In the 3D chart of Fig. 5, O_B is the center of the camera's lens, and also is the origin of frame B. Distance between O_B and the center of the image plane is f, which is the camera's focal length. In the 2D chart of Fig. 5, O_I is the origin of image frame, and the coordinate of image center O_0 is (u_0, v_0) with respect to the image frame and $(0, 0, f)$ with respect to the body frame. The coordinate of point P_0 is (u, v) with respect to the image frame and (x_0, y_0, f) with respect to the body frame.

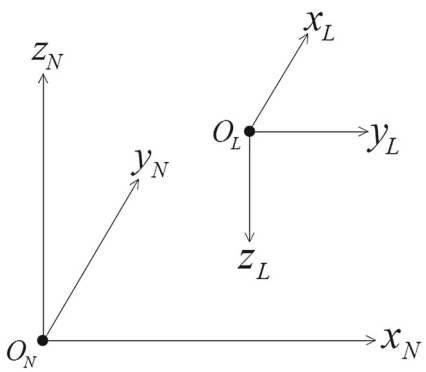


Fig. 3 Inertial frame and local reference frame

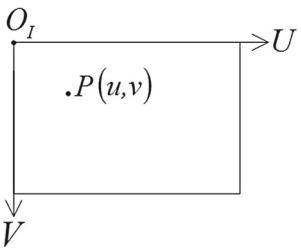


Fig. 4 Image frame

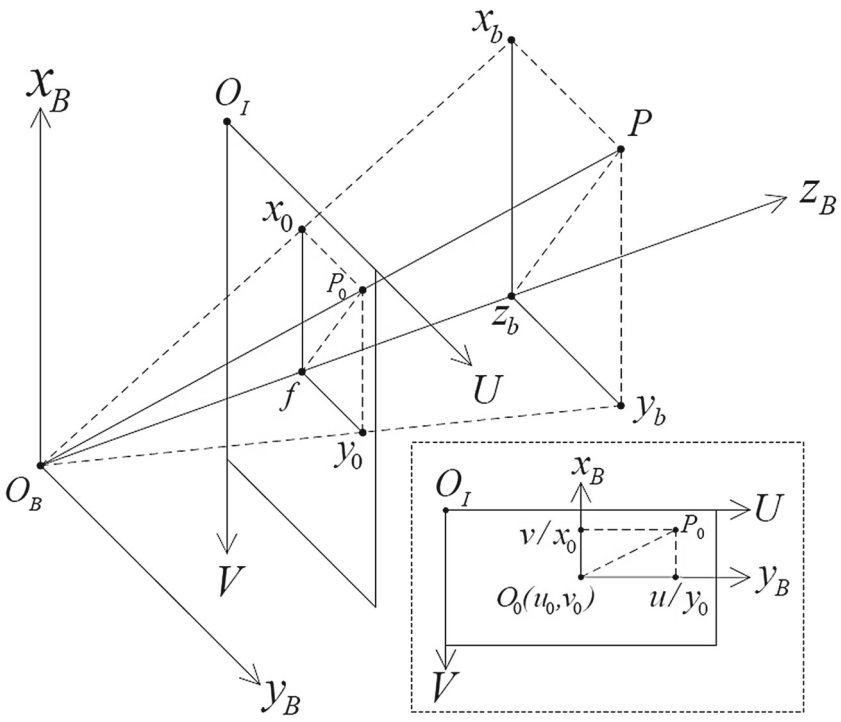


Fig. 5 Relationship between image frame (respect to the bottom camera) and body frame

Now we define rotation matrices for the 3D frames. Let $\overrightarrow{p_B}$ be the vector respect to frame B, where $\overrightarrow{p_B} = \begin{bmatrix} x_b & y_b & z_b \end{bmatrix}^T$. And let C_B^L be the rotation matrix which transfers the vector respect to frame B to the vector respect to frame L. So $\overrightarrow{p_L} = C_B^L \overrightarrow{p_B}$. Also, by chain rule, $C_B^N = C_B^L C_L^N$. From Fig. 3, we have the rotation matrix

$$C_L^N = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{5}$$

Rotation matrix C_B^L can be generated from sin and cos functions of Euler angles. We set ϕ be the roll angle, θ be the pitch angle, and ψ be the yaw angle. We also denote $s\theta$ for $\sin\theta$ and $c\theta$ for $\cos\theta$. Then we have

$$C_B^L = \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta s\psi & c\phi c\psi + s\phi s\theta s\psi & -s\phi c\psi + c\phi s\theta s\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{6}$$

From Fig. 5, we can derive the transformation from image frame I (respect to the bottom camera) to body frame B. Suppose a light source, located at point P, sends light to point O_B through point P_0, where P_0 is located at the image plane. We have

$$u = \frac{y_0}{k} + u_0 \tag{7}$$

$$v = -\frac{x_0}{k} + v_0 \tag{8}$$

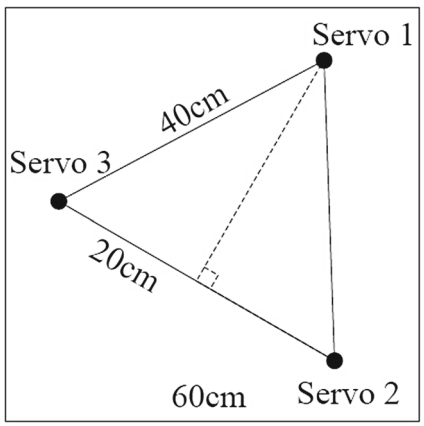


Fig. 6 Servo distribution on the bottom of landing plate

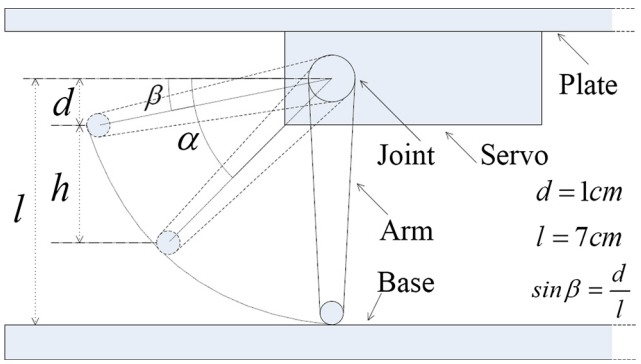


Fig. 7 Servo point configuration

where k is the size of the pixel, which is a constant parameter of the bottom camera, and the unit of k is meter. We also have

$$\frac{x_0}{x_b} = \frac{f}{z_b} \tag{9}$$

$$\frac{y_0}{y_b} = \frac{f}{z_b} \tag{10}$$

From Eqs. 7 to 10, we have

$$x_b = -\frac{k}{f}(v - v_0)z_b \tag{11}$$

$$y_b = \frac{k}{f}(u - u_0)z_b \tag{12}$$

The constant $\frac{k}{f}$ is determined by experiments using the least square method [24]. We have obtained $\frac{k}{f} = 0.00146$.

3 Vessel Deck Emulator

We have constructed a moving vessel deck emulator as the testing platform. The detailed design and the programed movement of the vessel deck emulator are introduced in this section.

3.1 Vessel Deck Design

Figure 6 shows the distribution of servos on the square plate for landing. The length of each side of the square is 60 cm. Servos are located at three vertices of an equilateral triangle, and the length of each edge is 40 cm.

Since the three independent servos share the same configuration, Fig. 7 only shows one of them for illustration. The height of this servo point is defined as h. Distance from the joint to the bottom of the servo d is measured as 1 cm, and length of the rigid arm l is measured as 7 cm. The angle between the landing plate and the arm α is programmable, and h changes when α changes. From the geometric relationship shown in Fig. 7, we have

$$h = l \sin\alpha - d \tag{13}$$

Since h reaches the maximum value of 6 cm when $\alpha = \frac{\pi}{2}$, and reaches the minimum value of 0 cm when $\alpha = \beta$, α is constrained as $\beta \leq \alpha \leq \frac{\pi}{2}$.

Under this configuration, from calculation, the landing platform has the capability of a maximum 6 cm heave movement and a maximum 10° pose angle movement (pose angle is the angle between the plate plane and the bottom base plane). Suppose that the vessel is 30 meters wide, from the geometric relationship, if the slope of deck is 10°, the minimum wave height is 5.2 m ($\frac{5.2}{30} \approx sin(10°)$). According to [4], this condition is under sea state 6, which is the "very rough" level.

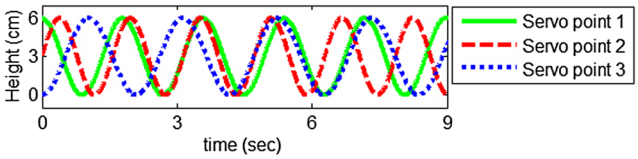


Fig. 8 Heignt control to each servo point

3.2 Deck Control and Results

We program the landing platform to emulate the deck motion under unfavorable sea state condition. Sinusoidal function is commonly used for simulating the movement of a vessel deck [4]. For our landing platform, we program the height of each servo point with a sinusoidal function independently. See Fig. 8.

Figure 9 is the actual motion of the landing platform under the control commands, and the data is an extracted 8 seconds' segment of a test from Vicon. The top plot is the position of the center of 'H' sign. The bottom plot is the attitude of the landing plane. From Fig. 9 we can see that the maximum altitude of the 'H' sign is 6 cm, and the maximum absolute roll and pitch angles of the landing plane are 10°. According to the analysis in Section 3.1, this movement emulates the deck motion of a vessel which is 30 meters wide under sea state 6 (very rough).

4 Long Range Landing Platform Detection and Tracking

For successful autonomous landing, the first task is to detect the landing platform. After detection, the drone should be able to track and approach to the platform autonomously. This section presents the method we used to achieve this goal.

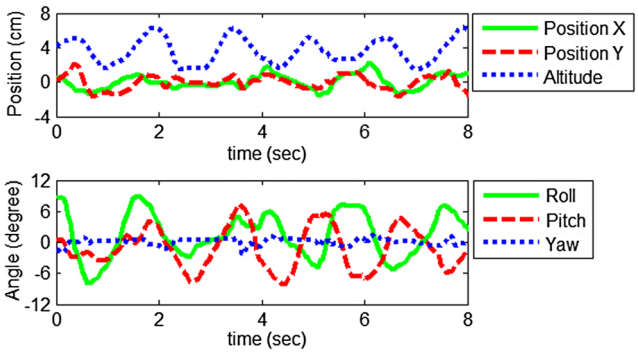
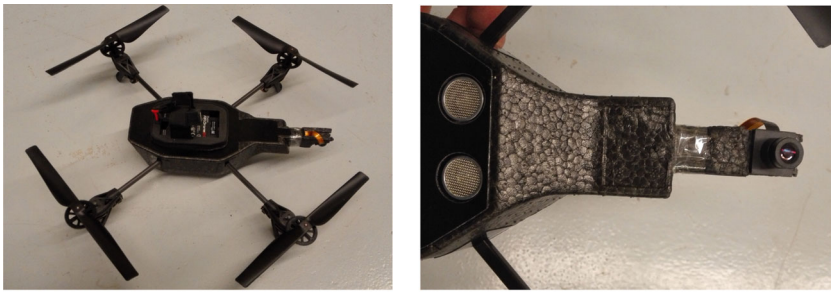
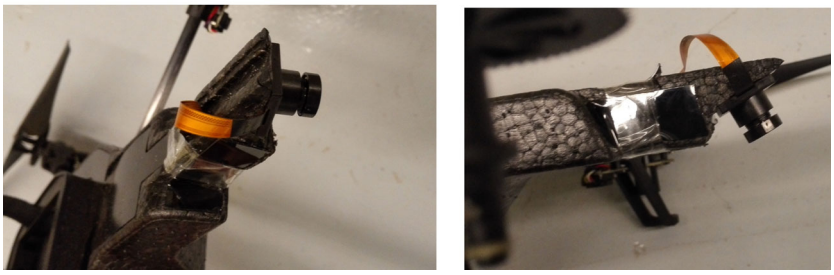


Fig. 9 The motion of the landing platform



(a) Top view of the front camera. (b) Bottom view of the front camera.



(c) Front view of the front camera. (d) Side view of the front camera.

Fig. 10 Front camera configuration after the modification

4.1 Front Camera Configuration

The front camera of AR.Drone is originally set to look forward, and as we discussed in Section 2.1, it has an approximate 60° horizontal FOV, and an approximate 30° vertical FOV. However, if this configuration is used without any adjustment, even if the algorithm can drive the drone to approach the landing platform, there will be a blind zone since the landing platform will not show in neither the view of front camera nor the view of the bottom camera. To eliminate the blind zone, we did some modification on the front camera. First, we changed the fix angle of the camera. It points along the X axis of body frame B before. After the modification, it points a little downward, and the central view respect to B is in X-Z plane, with about 75° of angle to X axis. Second, we rotated the camera by 90°. So it has an approximate 60° vertical FOV, and an approximate 30° horizontal FOV. Figure 10 shows the front camera configuration after the modification.

Figure 11 illustrates the entire camera configuration after the modification. In Fig. 11, α is the vertical FOV of the front camera, and β is the angle looking backward of the front camera. The distance between the drone and the landing platform is notated as l. The coordinate of the landing platform in the image frame is (u, v), and since it rotated by 90°, in side view, only u is shown there.

From this configuration, if the drone has enough altitude h, there will be an overlap FOV area between the front camera and the bottom camera. And once the landing platform appears in the overlap area, the system switches to bottom camera mode autonomously and it won't have a blind zone.

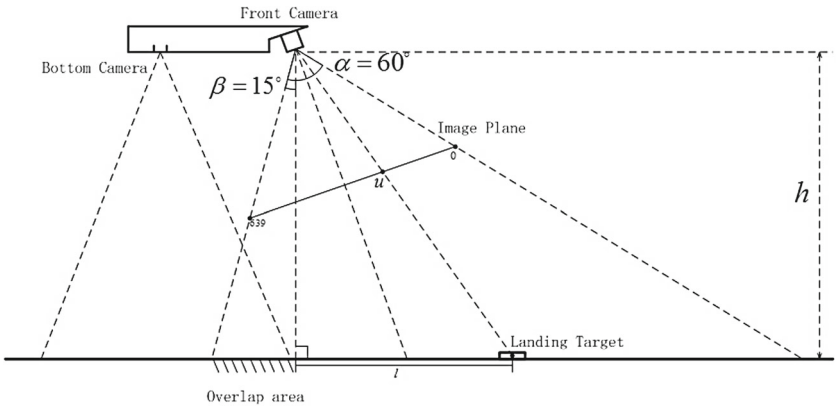


Fig. 11 Camera Configuration

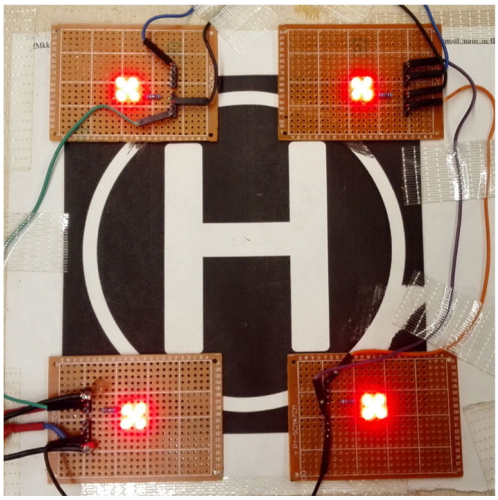


Fig. 12 Red LEDs and 'H' landing sign configuration

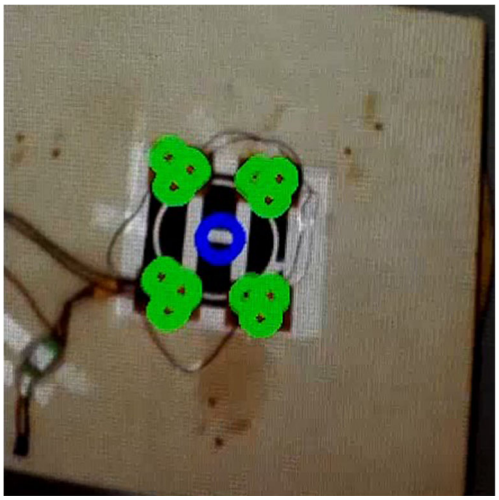


Fig. 13 Red LEDs recognition and landing platform calculation

4.2 Landing Platform Recognition

We fixed 4 red LEDs at the 4 corners of the landing 'H' sign. As we introduce in Section 2.1.2, these LEDs are used for the UAV to search the landing platform from a long range. See Fig. 12. We note this is a simplified model of the reality. For long range shipboard detections, the ship can be seen as a single hot spot by the on-board IR camera [25]. Limited by the AR. Drone's cameras and indoor environment, we use red LEDs to simulate the hot spot and use the front camera of the AR. Drone to recognize and track the red LEDs.

In this part, red dots recognition algorithm is designed and used. The algorithm first convert the image into 3 matrices to store the RGB values. A simple but effectively algorithm is used. The dot which has the R value greater than 150, G and B values are smaller than 80 is considered as the red dot. After the algorithm searches over the entire image, a matrix is generated to store the coordinates in the image frame of the red dots. Then the average of the coordinates is calculated, which is considered as the coordinates of the landing platform. Figure 13 shows the recognition results. The recognized red dots are circled with green circles, and the calculated average coordinate (landing platform coordinate (u, v)) is circled with a blue circle. We note the current algorithm can't distinguish between lights belonging to the landing pad and others, and we don't have other red light sources in the experiment area. This limitation will be studied in future.

4.3 States Estimation

After we get the coordinate of the landing platform in the image frame, the next step is to estimate the states of the drone. As we introduce in Section 2.1.1, the altitude z, roll angle ϕ and pitch angle θ are directly available from AR.Drone. The goal here is to estimate the heading angle and the horizontal distance between the drone and the landing platform.

4.3.1 Heading Angel Estimation

The heading angle we discussed here represents the angle between the line connecting the center of the landing platform and the projection of the drone on ground, and the projection of the x axis of the body frame B (x_B). Coordinate v of the landing platform in the image frame is used to determine the heading angle ψ. Because the horizontal FOV we measured is 30°, and the pixel number is 360 in that direction, we assume that $v = 359$ and $v = 0$ represent ±15° heading angle, and $v = 180$ represent 0° heading angle. Furthermore, the relation between coordinate v and heading angle ψ is assumed as linear. Then we get the heading angle estimation represented in Eq. 14.

$$\psi = \frac{v - 180}{12} * \frac{\pi}{180} \tag{14}$$

4.3.2 Horizontal Distance Estimation

Coordinate u of the landing platform in the image frame is used to determine the horizontal distance l (assuming the yaw angle $\psi = 0$). Refer to Fig. 11. The vertical FOV is 60° (α), and there is a 15° backward view (β). The pixel number in that direction is 640. Since we know altitude h and coordinate u, from geometric relationship, we can calculate l. First, we define angle θ to be the angle between the optical line connecting the lens' center and the landing platform, and the main axis of the lens. We get

$$tan\theta = \frac{320 - u}{w} \tag{15}$$

where w is equivalent pixel number of the distance between the center of the image plane and the center of lens.

We also have

$$tan\frac{\pi}{6} = \frac{320}{w} \tag{16}$$

Combining Eqs. 15 and 16, we have

$$\theta = arctan\frac{(320-u)*tan\frac{\pi}{6}}{320} \tag{17}$$

If θ is greater than $-15°$, the landing platform is in front of the drone, and

$$l = h*tan(\theta + 15*\frac{\pi}{180}) \tag{18}$$

If θ is smaller than $-15°$, the landing platform is in back of the drone, and

$$l = -h*tan(\theta + 15*\frac{\pi}{180}) \tag{19}$$

4.4 Approach Procedure

In the stage of approaching from long range, the goal is to drive the drone to an area above the landing platform. Once the approach process completes, the drone will switch to the bottom camera autonomously, and the landing 'H' sign will be located in the FOV of the bottom camera. To achieve this goal, a robust approaching procedure is designed and introduced in this section. Since we already have the state estimation, the control algorithm is also introduced.

4.4.1 PID Control Law

A PID control method for discrete time systems is used in this work. In general, let $s(k)$ be the state measured in step k, $s^d(k)$ be the desired state of step k, then the error at step k is

$$s^e(k) = s^d(k) - s(k) \tag{20}$$

The summation of historical errors at step k is

$$s^{et}(k) = \sum_{i=0}^{k} s^e(i) \tag{21}$$

The difference of error at step k is

$$s^{ed}(k) = s^e(k) - s^e(k-1) \tag{22}$$

Then the control input u at step k is defined as

$$u(k) = Ps^e(k) + Is^{et}(k) + Ds^{ed}(k) \tag{23}$$

Where P, I, D are the gains of the PID controller.

For the control of heading angle ψ and roll angle ϕ, the goal is to maintain them equal to 0°. The measured angles are the control error inputs, since the goal is to drive them to 0°. If the heading angle equal to 0°, it means that the head of the drone is pointing to the landing platform. Under that condition, we can change the pitch angle θ to drive the drone to approach to the landing platform. For the control of altitude z and pitch angle θ, the desired value is changing during the approaching, and the detailed procedures are presented next.

4.4.2 Approach Process

First, the drone takes off from an arbitrary position in the lab, hovering at 2 meters. Then it turns around without position change to find the red dots. After the red dots appear in the view of the front camera, the algorithm locks the red dots, and calculates the relative heading angle. Heading angle PID controller adjusts the heading angle around 0°. Under this condition, the drone changes its pitch angle to approach the landing platform. Notice that the drone approaches the landing platform only when the heading angle is small. If the angle is not small, the drone is programed to stop moving forward and hover at the current position, while the heading angle PID controller will drive the heading angle to 0°.

When approach, the red dots' position in image is also changing. To make sure the drone can always see the red dots, an altitude PID controller drives the drone to decrease its altitude if the coordinate u of the landing platform is too close to the image boundary during the approach process. Once the altitude reaches to 1 meter, the altitude PID controller stabilize the height to be 1 meter without further descending. This is because at 1 meter altitude, the view of the bottom camera is clear and wide enough for later landing. During this process, the horizontal distance l is calculated by using Eqs. 18 and 19.

Finally, when l becomes negative, which means the landing 'H' sign is in the overlap FOV area between the front camera and the bottom camera, the algorithm switches to the bottom camera mode autonomously, and the landing stage algorithms start to work. Figure 14 is the flow chart of the designed approach process.

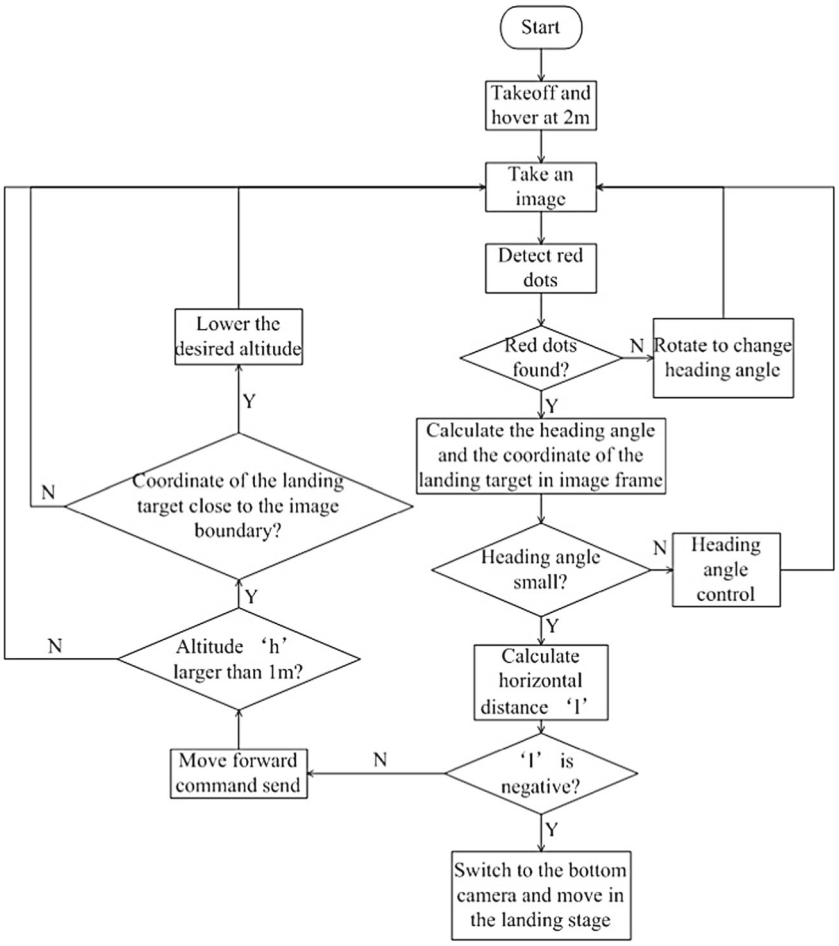
5 Hovering and Landing

After the drone hovers above the landing area, the second stage starts and the goal is to land on the 'H' sign accurately. This section presents our methods to achieve this goal.

5.1 Image Processing

'H' landing sign is chosen as the landing beacon due to its widely used. The goal for the image processing is to find the 'H' sign in the image and locate the center of the sign in the image frame in real-time. Image moments technique is used for the 'H' sign tracking. In image processing, image

Fig. 14 Flow Chart Illustration



moment usually contains some special characteristics of the contour, and image moment is generally used to recognize certain objects or patterns [5].

Before the flight, the contour of a standard 'H' is extracted from a reference image. During the flight, 'H' sign tracking algorithm keeps comparing the moment of each qualified contour derived from each image frame with the standard moment of 'H' contour. If the similarity of the most similar contour is within a selected threshold, the contour will be recognized as the 'H' landing sign. In this part, OpenCV library functions are used for contour extraction and comparison. Details are presented next.

Fig. 15 Reference image

Fig. 16 Contour of a standard H

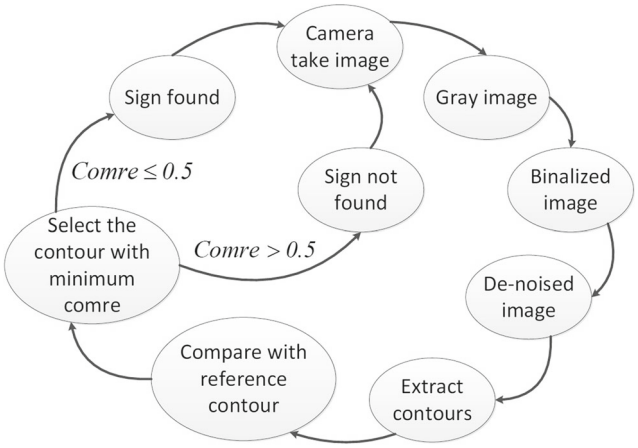


Fig. 17 State machine of the image processing

5.1.1 Standard 'H' Contour Extraction - Before the Flight

Figure 15 is the reference image we used for standard 'H' contour extraction. We also printed it out as the 'H' landing sign on the landing plate. The reference image is a binarized image. After we load it in OpenCV environment, function 'findContours' is called to extract all contours. We select and save the contour of 'H', and use red line to draw the 'H' in Fig. 16.

5.1.2 Real-time 'H' Sign Tracking - During the Flight

During the flight, the bottom camera will capture color images in 20Hz. For each image, it will first be converted to a gray image. Furthermore, the gray image will be converted to a binarized image by carefully selecting a threshold. We can do this because the 'H' sign has high contrast with the background. The binarized image will be used to extract contours. Before we extract the 'H' sign's contour, a de-noising process will be conducted. After that, we extract all contours detected in the image, and eliminate the contours which are too small to be the 'H' contour by calculating the contour area. Then the contours left will be compared with the standard 'H' contour saved before. OpenCV function 'matchShapes' is used for the comparison. The function first calculates the moment of contour, and then compares it with the standard 'H' contour moment. The result is given as 'comre', which represents the similarity of them. The smaller the 'comre' is, the higher similarity they have. A threshold for 'comre' is set to be 0.5. In each image frame, the contour which has the minimum 'comre' with the standard 'H' contour, which is also within the threshold, will be selected as the 'H' landing sign. Figure 17 is the state machine of the image processing.

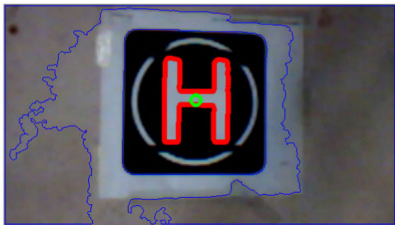
Figure 18 shows the operation results during the image processing. Figure 18a is the original color image taken from the bottom camera. Figure 18b is the gray image converted from Fig. 18a. Figure 18c is the binarized image after de-noising. Binalize makes it easier to find contours, and de-noising eliminates the small dots, which decreases the number of invalid contours and reduces the burden on comparing algorithms. Figure 18d shows a case that 'H' landing sign is detected. Once the algorithm finds the 'H' landing sign, the contour of 'H' will be drawn. Then, OpenCV function 'minAreaRect' is used to bound the 'H' with a minimum area of rectangular, and also to provide the center coordinate in the image frame I. Because 'H' is

Fig. 18 Operation results during the image processing

(a) Original color image from the bottom camera.

(b) Gray image.

(c) Binarized image with de-noising.



(d) 'H' landing sign detected.

symmetric, this coordinate is also the coordinate of the 'H' landing sign. After getting the center coordinate, we draw a small circle at the center.

Because the landing 'H' sign will move with the deck emulator, it will not always be parallel to the X-Y plane of the body frame. To further test the robustness of the proposed 'H' sign detection algorithm, we take pictures of the 'H' sign with different orientations and distances, using different cameras under different lighting conditions to test if the algorithm can still work. Figure 19 shows some of the test results.

In Fig. 19, the 'Bright light condition' is obtained by using a computer screen to show the 'H' landing sign, the 'Medium light condition' is obtained in our lab with all lights turned on, the 'Dark light condition' is obtained in our lab with only half lights turned on. The 'Smartphone camera' we used has 1920×1080 resolution, while the AR.Drone bottom camera has 640×360 resolution. The 'Medium distance' in this test is 60 cm, while the 'Short distance' in this test is 30 cm. In Fig. 18d, the distance from the sign and the camera is 1m.

From these experiments, we conclude that the 'H' sign detection algorithm is robust. It can work under different lighting conditions, different resolutions of camera, different distances, different orientations and different tilt angles. Additionally, we tested the maximum tilt angle of the 'H' when this algorithm works using another experiment, and found that the maximum tilt angle is 20°. Since the maximum pose angle of the designed deck emulator is 10°, the proposed algorithm can track the 'H' landing sign successfully when the landing platform is moving.

5.2 State Estimation

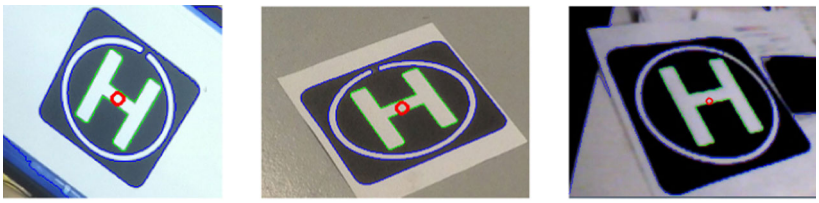
Only on board sensors are used for state estimation. Sensors provided by AR.Drone are a 6 degree of freedom internal IMU, an ultrasonic altimeter, and a forward-looking (front) camera, and a downward-looking (bottom) camera.

Six states are required for autonomous landing, including 3 components of position vector with respect to local reference frame (x_l, y_l, z_l), and 3 Euler angles (ϕ, θ, ψ). In this section, we present how to calculate these states by using sensors mentioned above. Notice we only use the bottom camera during the landing stage.

Two Kalman filters are designed to improve the estimation accuracy. States for the first Kalman filter are z_l, ϕ, and θ, which are always available. And states for the second Kalman filter are x_l, y_l, and ψ, which are available only when ' H' landing sign is detected.

From AR. Drone system, users can directly acquire altitude z_l, roll angle ϕ, and pitch angle θ, where z_l is calculated using ultrasonic altimeter, and ϕ and θ are calculated using the IMU.

Although AR.Drone also provides users with the yaw angle ψ, the yaw angle drifts significantly. Since we don't use GPS or other external positioning systems, we can't



(a) Bright light condition; Smart-phone camera; Medium distance.

(b) Medium light condition; Smart-phone camera; Medium distance.

(c) Medium light condition; AR.Drone bottom camera; Short distance.

(d) Dark light condition; AR.Drone bottom camera; Medium distance.

(e) Dark light condition; AR.Drone bottom camera; Medium distance.

Fig. 19 'H' sign tracking algorithm robustness test

acquire position information directly. Solutions for x_l, y_l, and ψ estimation are obtained by using images once the 'H' landing sign is recognized.

After we bound the 'H' sign with a minimum area of rectangular by calling the OpenCV library function 'minAreaRect', which is the last step when the 'H' landing sign is recognized in the image processing, the angle between this rectangular's long side and the image's long side is also provided in the output of the function 'minAreaRect'. According to the 2D chart shown in Fig. 5, this angle is between 'H' landing sign's vertical axis and the x_B axis of body frame. We denote the angle as γ, therefore γ is the relative heading angle between the drone and the landing platform. We denote the angle of 'H' landing sign's vertical axis and north direction (x_L axis) as δ, so δ is the setting angle of the landing platform respect to the inertial frame. Then we have

$$\psi = \gamma + \delta \tag{24}$$

In our experiments, we set $\delta = 0$, and then the relative heading angle γ is equal to the yaw angle ψ. We note that since the 'H' shape used for the landing pad is symmetric, the heading estimation will have 180 degrees of uncertainty. In real applications, we shall use other information to eliminate that uncertainty. For example, when approaching from the long range, the heading direction of the ship could be determined, based on which a unique heading angle γ can be determined.

For position vector $\overrightarrow{p_L}(x_l, y_l, z_l)$ estimation, we have

$$\overrightarrow{p_L} = C_B^L \overrightarrow{p_B} \tag{25}$$

Once we obtain the values of ϕ, θ, and ψ, rotation matrix C_B^L is determined. Combining Eq. 25 with Eqs. 11 and 12, we rewrite Eq. 25 as

$$\begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} -\frac{k}{f}(v - v_0) \\ \frac{k}{f}(u - u_0) \\ 1 \end{bmatrix} z_b \tag{26}$$

In Eq. 26, the unknown components are x_l, y_l, and z_b. By expanding Eq. 26, the third row becomes

$$z_l = \left\{ s\theta \left[\frac{k}{f}(v - v_0) \right] + s\phi c\theta \left[\frac{k}{f}(u - u_0) \right] + c\phi c\theta \right\} z_b \tag{27}$$

Once we acquire z_b from Eq. 27, $\overrightarrow{p_B}$ and $\overrightarrow{p_L}$ can be calculated.

5.3 Landing Controls

AR.Drone de-couples the controls of motors' speed into 4 independent controls of roll and pitch angles, yaw angle rate and altitude rate, while positions are controlled by controlling the roll and pitch angles. The PID controller introduced in Section 4.4.1 is used.

For hovering and landing on the 'H' sign, the drone is required to stay at a position above the sign. The errors of positions are calculated respect to the body frame, and the desired position $({x_b}^d, {y_b}^d)$ is $(0, 0)$. Thus the errors of positions are ${x_b}^e = -x_b$ and ${y_b}^e = -y_b$.

We also require the forward direction of the drone parallel to the 'H' sign when landing. So the desired value of γ is 0 and the error is ${\gamma}^e = -\gamma$.

The error of altitude is ${z_l}^e = {z_l}^d - z_l$, where the desired altitude ${z_l}^d$ is designed a prior. An altitude planner is designed to calculate ${z_l}^d$. During the hovering phase, ${z_l}^d$ is set as a constant c (c is positive). During the landing phase, it's set to be a function of time t, such as ${z_l}^d = -qt + c$ (q is also positive, and used to control the descending rate).

Sensor data update rate of AR.Drone is 20 Hz. For each update, control commands are also sent to AR.Drone. Figure 20 shows the framework of the autonomous flight algorithms during the landing stage.

6 Experimental Results

6.1 Approaching From Long Range

To test the long range approach, we put the landing platform in the center of the flight area (Coordinate of the landing platform in the inertial frame is $(0, 0)$), and initialize the drone 2 m from the landing platform with random initial heading directions. The length of our flight area is 5 m, and the width is 3 m. For each experiment, we choose different initial positions of the drone to take off to test the performance. Using the designed procedure presented in Section 4.4.2, we did 20 independent experiments. The

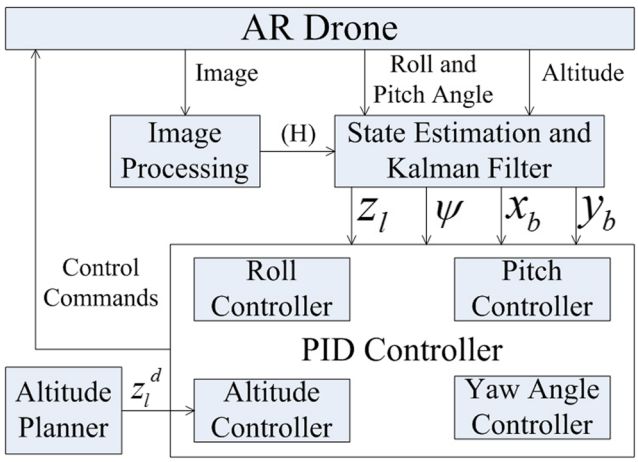


Fig. 20 Framework of the autonomous landing algorithms

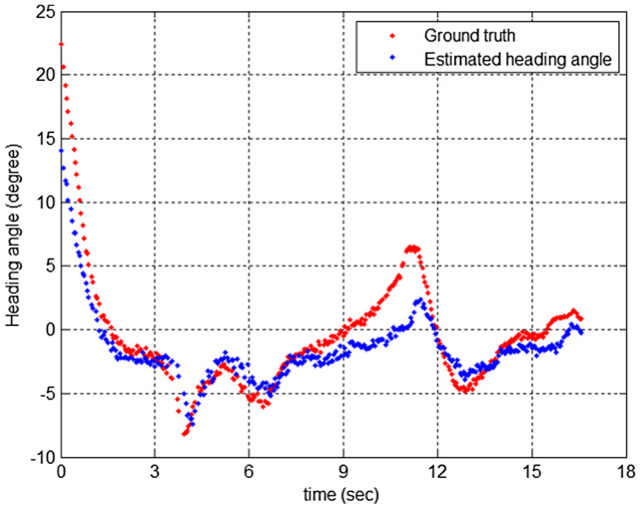


Fig. 21 Heading angle estimation and control results

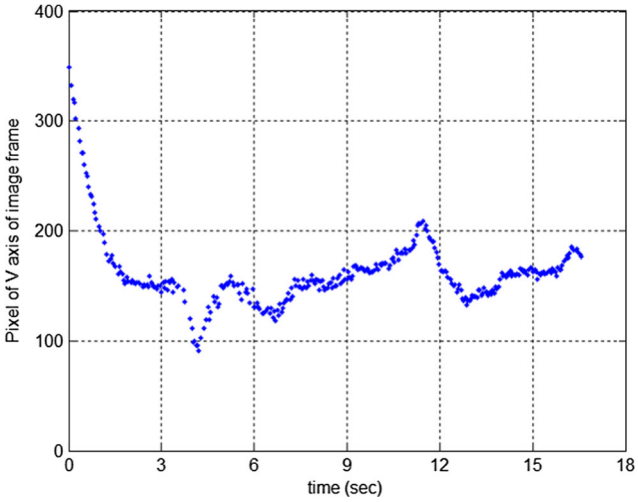


Fig. 22 Coordinate 'V' of the landing platform in image frame

success rate of these experiments is 100%, and the average approaching time is 20s.

Here we use one experiment to illustrate the approaching stage. In this experiment, the drone takes off at the bottorm-right of the lab (Coordinate of the drone in the inertial frame is $(0.9, -2)$ when the drone starts to approach). The time $t = 0$ is the time when the red dots on the landing platform start to appear in the view of the front camera. At time $t = 16.5$, the approach completes, and the drone hovers above the landing platform.

6.1.1 Heading Angle Estimation and Control

As presented in Section 4, the key factor for successful tracking and approaching to the landing is accurate heading angle estimation and control. Based on Section 4.3.1, the coordinate V in the image frame of the calculated landing sign is used to calculate the heading angle. See Eq. 14.

Figure 21 is the heading angle estimation and control results. At $t = 0$, the red dots on the landing platform start to appear in the view of the front camera. The estimated heading angle is about 15°, and the actual heading angle obtained by the Vicon system is 22.5°. After that, the heading angle PID controller drives the heading angle to 0°, and tries to keep the heading around 0°. We notice that at 4 seconds and 11 seconds, there are two heading angle peaks. However, the heading angle drift will not cause approach failure. As presented in Section 4.4.2, the drone only change its pitch angle to approach when the heading angle is small (within ±5°). If the absolute heading angle is larger than 5°, the drone will first suspend approaching and correct the heading angle.

Figure 22 shows the calculated coordinate V of the landing platform in the image frame during the approach. We noticed that the trend in Fig. 22 is similar to the estimated heading angle in Fig. 21, which is because the estimated heading angle is calculated using Eq. 14.

We define the estimation error as the difference between the ground truth and the estimated value during the approaching stage. In this experiment, the average estimation error during this flight is 1.03°, the standard deviation of the estimation error is 1.91°, and the maximum absolute error estimation is 8.35°. We conclude that our heading angle estimation algorithm is suitable. According to the ground truth, after the yaw angle becomes stable at 3s to the end of the test, the average heading angle is −1.29°, the standard deviation of the heading angle is 3.04°, and the maximum absolute heading angle is 6.49°. We conclude that our heading angle control algorithm is suitable.

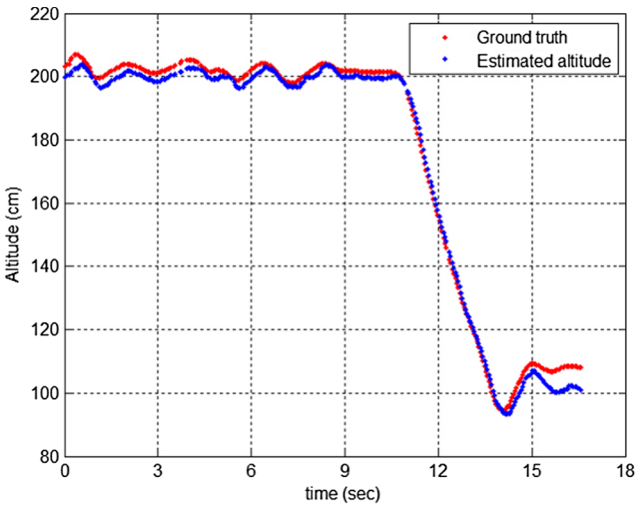


Fig. 23 Altitude estimation and control results

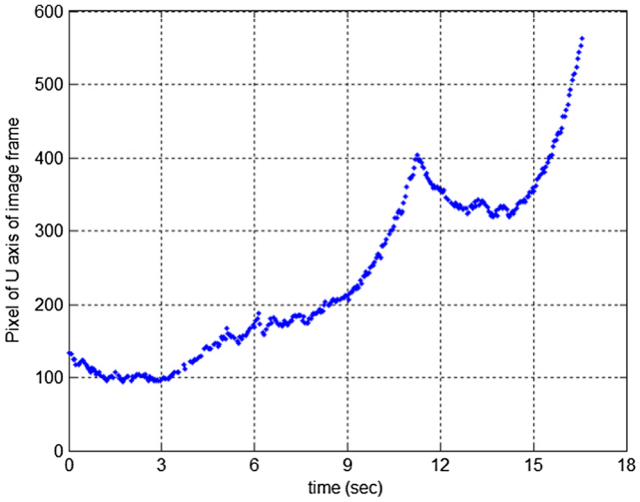


Fig. 24 Coordinate 'U' of the landing sign in image frame

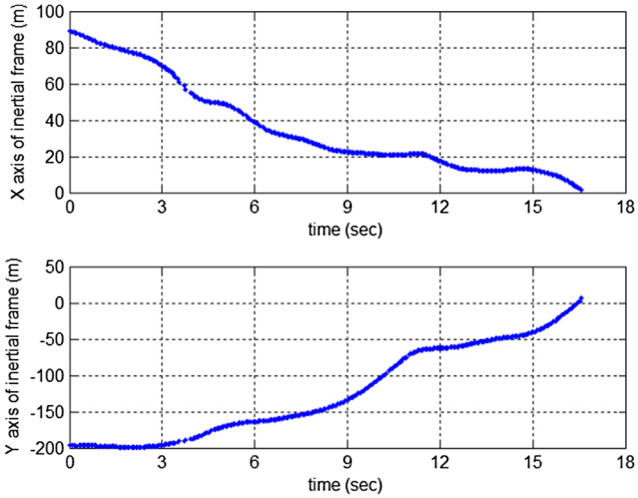


Fig. 25 Position change during the approaching

6.1.2 Altitude Change During Approach

The altitude of the drone is designed to descend during the approach process. As presented in Section 4.4.2, the initial hovering altitude is 2 meters, and for hovering above the landing sign, the altitude is 1 meter. The advantage of altitude descending is to keep the landing platform in the FOV of the front camera. Figures 23 and 24 show the experimental results of the altitude change during the approaching.

From Fig. 23, we see that the drone starts to descend at 11s, and stops descending at 14s. After 14s the drone keeps its altitude at 1m, and the oscillation around 15s is the overshoot of the altitude PID controller. From Fig. 24, we see that when the drone finds the red dots sign, the initial calculated U coordinate of the sign in the image frame is 120. During the approach process, the U coordinate increases, and reaches 400 at 11s. If the altitude stays at 2 meters, and drone continues approaching, the U coordinate will continue increasing. Thus, the landing platform will be out of the FOV of the front camera before the drone hovers above it. Because at 11s, the altitude starts to descend, the U coordinate starts to decrease. The altitude stops decreasing at 14s, and the U coordinate also stops descending. After that, the U coordinate increases again, and reaches 570. At this moment, the sign can be seen in the bottom camera of AR.Drone, the algorithm switch to the bottom camera, the approaching stage ends, and the landing stage begins.

6.1.3 Approaching Trajectory

Figures 25 and 26 show the trajectory of the approaching. The position is captured by Vicon system. In Fig. 25, we can see that the initial position of the AR.Drone is $(0.9, -2)$ in the inertial frame, and reaches $(0, 0)$ when the approach process ends. We noticed that at $t = 11s$, the drone suspends to approach, which is because the heading angle is more than 5°. We can see that the coordinate keeps at $(0.2, -0.6)$ from 11s to 12s. A 2D trajectory is plotted in Fig. 26.

6.2 Experiment of Autonomous Hovering

6.2.1 Hover Above a Stationary Platform

In this experiment, the landing platform is stationary and AR.Drone is required to hover above the 'H' landing sign autonomously. Figure 27 shows the autonomous hovering experiment results. The green dash is the ground truth (data from Vicon motion capture system) and the red dot is the estimated results. Table 1 is the summarized statistical results of the estimation errors during 60 seconds. Positions are given in cm, and Euler angles are in degrees.

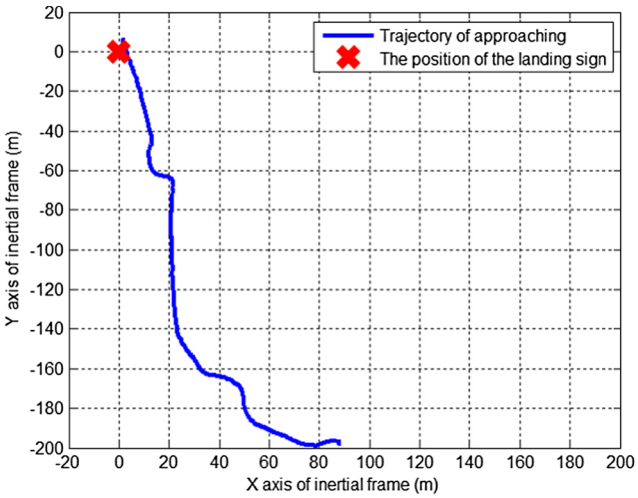


Fig. 26 Trajectory of the approaching

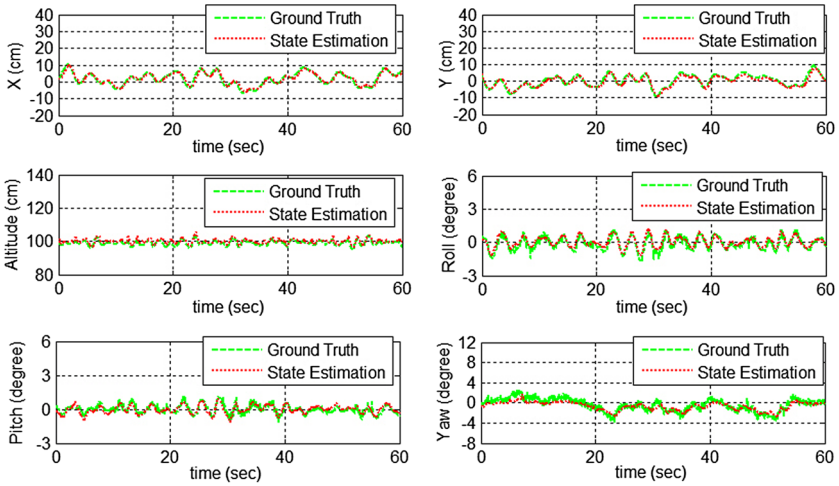


Fig. 27 Hover test results when landing platform is stationary

Table 1 Evaluation of estimation performance when landing platform is stationary

	$x_l(cm)$	$y_l(cm)$	$z_l(cm)$	$\phi(^\circ)$	$\theta(^\circ)$	$\psi(^\circ)$
Average error	−0.05	−0.44	1.07	0.13	−0.02	−0.28
Error standard deviation	0.95	1.01	0.82	0.22	0.24	0.63
Max. absolute error	2.44	2.95	4.21	1.12	0.99	2.05

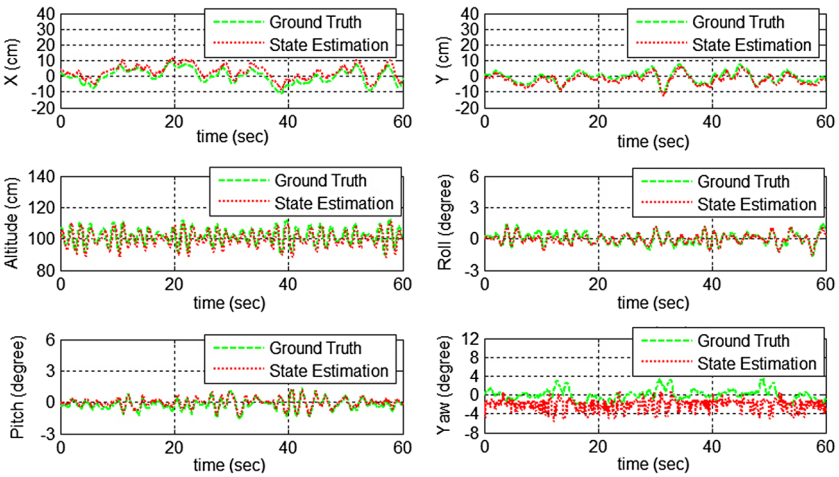


Fig. 28 Hover test results when landing platform is moving

Table 2 Evaluation of estimation performance when landing platform is moving

	$x_l(cm)$	$y_l(cm)$	$z_l(cm)$	$\phi(^\circ)$	$\theta(^\circ)$	$\psi(^\circ)$
Average error	2.75	−1.30	−1.74	−0.14	0.15	−2.14
Error standard deviation	1.34	1.21	3.15	0.23	0.35	1.82
Max. absolute error	7.43	4.72	8.55	1.73	1.56	2.73

From the test results, we can see that the average error of the position estimation is less than 2 cm, and the average error of the Euler angle is less than 0.3°. The standard deviation of the position estimation error is around 1 cm, and the standard deviation of the Euler angle estimation error is less than 0.7°. The maximum absolute position estimation is less than 5 cm, and the maximum absolute Euler angle estimation error is less than 3°. We can also see that the problem of yaw angle drift is eliminated once it is estimated from the image.

6.2.2 Hover Above a Moving Platform

In this experiment, the landing platform has pose and heave motion while the drone hovers above the 'H' landing sign autonomously. Figure 28 shows the test results. The green dash is the ground truth (data from Vicon motion capture system) and the red dot is the estimated results. Table 2 is the summarized statistical results of the estimation errors. Positions are given in cm, and Euler angles are in degrees.

From the test results, we can see that the average error of the position estimation is less than 3 cm, and the average error of the Euler angle is less than 3°. The standard deviation of the position estimation error is less than 3 cm, and the standard deviation of the Euler angle estimation error is less than 2°. The maximum absolute position estimation is less than 9 cm, and the maximum absolute Euler angle estimation error is less than 3°.

Comparing with the results on the stationary landing platform, we observe two aspects. First, oscillation of the altitude is larger, which is due to the heave movement of the landing platform. The ultrasonic altimeter measures the distance from the drone to the surface of the platform. Second, because of the movement of the landing platform, the performance of hovering this time is slightly poorer compared to the test when the landing platform is stationary. The average error of both position estimation and Euler angle estimation, the error standard deviation of both position estimation and Euler angle estimation, and the maximum absolute error of both position estimation and Euler angle estimation are slightly larger.

6.3 Experiments of Autonomous Landing

The difference between landing and hovering is that the desired altitude remains constant during hovering but changes during landing. In landing, an altitude planner will generate the sequence of desired altitudes, and the PID controller will drive the drone to track the reference. At the same time, position and yaw angle PID controllers are also taking actions. Figure 29 shows the altitude when landing. In Fig. 29, blue dots represent the actual altitude, and red dots represent the desired altitude generated by the altitude planner. Before 25.4 s, the drone is hovering with oscillation due to the moving platform. The landing starts at 25.4 s and ends at 29.8 s. From 25.4s to 29.4s, the actual altitude is seen to follow the desired altitude. At 29.4 s, the actual altitude drops to 26 cm, and 'H' sign is no longer completely shown in the image, and the landing function of AR.Drone is called. The drone lands on the platform at 29.8 s, and after that it moves with the platform.

We further evaluate the landing accuracy through statistical tests. We first test on a stationary platform, and then test on a moving platform. For each situation, we did 20 independent experiments. In all experiments, the drone finds the landing platform and lands on it autonomously. Figures 30 and 31 show the landing accuracy results.

Figure 30 is the landing position results when the landing platform is stationary. The red circle is the center of 'H',

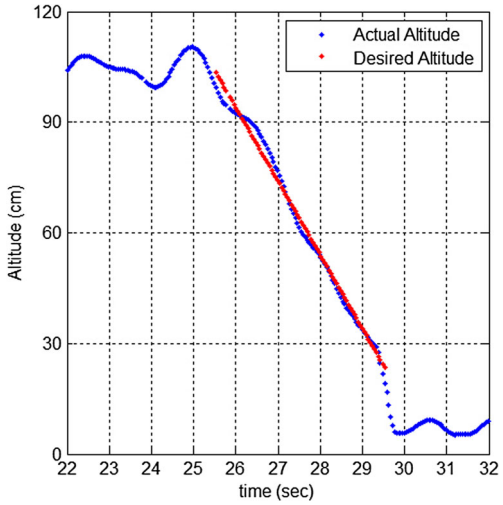


Fig. 29 Altitude landing performance

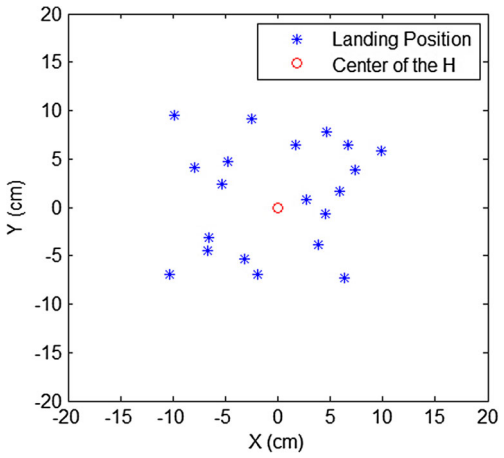


Fig. 30 Position accuracy evaluation when the landing platform is stationary

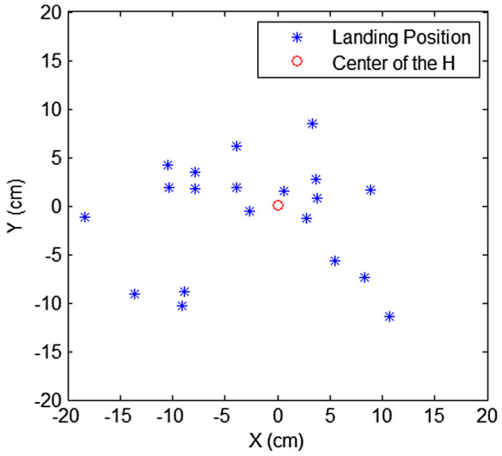


Fig. 31 Position accuracy evaluation when the landing platform is moving

and the blue stars are the landing positions from the trials. After AR.Drone lands on the platform, the distance from the body frame center to the center of 'H' sign is measured. The average distance error is 7.9 cm, and the standard deviation of the landing error is 2.64 cm. If we define success landing as the case that the drone completely lands on the moving platform, our landing success rate is 100%. Figure 31 is the landing position results when the landing platform is moving. The average distance error is 9.0 cm, and the standard deviation of the landing error is 4.79 cm.

Compare with the accuracy when the landing platform is stationary, this result is slightly poorer because of the movement of the landing platform, but the landing success rate is still 100%. Compare with [20], which did not define clearly the success landing, the average distance error is 12 cm under no environment disturbance, and the maximum success landing rate is reported as 90%, our results are significantly better.

7 Conclusion and Future Work

We present an autonomous approach for a quadrotor to land on a vessel deck from long range. On-board IMU and altimeter, image processing, and Kalman filters are used to provide the state estimation. Experiments are implemented on a self-designed landing platform, which can emulate the deck motion of a vessel 30 m wide under sea state 6. Flight results demonstrate the landing success rate of 100% with a 9 cm average distance error. In the future, we will resident the landing platform on a ground vehicle which provides additional horizontal motion of the deck. Additionally, wind disturbance will be considered to make the algorithms more robust.

Acknowledgements The authors would acknowledge the research support from the Air Force Office of Scientific Research (AFOSR) FA9550-16-1-0184 and the Office of Naval Research (ONR) N00014-16-1-2729. The instructive suggestions from Dr. David B. Findlay are also gratefully acknowledged.

References

1. Jin, S., Zhang, J., Shen, L., Li, T.: On-board vision autonomous landing techniques for quadrotor: a survey. In: Control conference (CCC), 2016 35th Chinese, pp. 10,284–10,289. IEEE, Chengdu (2016)
2. Ling, K., Chow, D., Das, A., Waslander, S.L.: Autonomous maritime landings for low-cost vtol aerial vehicles. In: 2014 Canadian conference on computer and robot vision (CRV), pp. 32–39. IEEE, Montreal (2014)
3. Daly, J.M., Ma, Y., Waslander, S.L.: Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays. In: 2011 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 4961–4966. IEEE, San Francisco (2011)
4. Sanchez-Lopez, J.L., Pestana, J., Saripalli, S., Campoy, P.: An approach toward visual autonomous ship board landing of a vtol uav. J. Intell. Robot. Syst. **74**(1–2), 113–127 (2014)
5. Gautam, A., Sujit, P., Saripalli, S.: A survey of autonomous landing techniques for uavs. In: 2014 international conference on unmanned aircraft systems (ICUAS), pp. 1210–1218. IEEE, Orlando (2014)
6. Pebrianti, D., Kendoul, F., Azrad, S., Wei, W., Nonami, K.: Autonomous hovering and landing of a quad-rotor micro aerial vehicle by means of on ground stereo vision system. Journal of System Design and Dynamics **4**(2), 269–284 (2010)
7. Kong, W., Zhang, D., Wang, X., Xian, Z., Zhang, J.: Autonomous landing of an uav with a ground-based actuated infrared stereo vision system. In: 2013 IEEE/RSJ international conference on intelligent robots and systems (IROS), pp. 2963–2970. IEEE, Tokyo (2013)
8. Kong, W., Zhou, D., Zhang, D., Zhang, J.: Vision-based autonomous landing system for unmanned aerial vehicle: a survey. In: 2014 international conference on multisensor fusion and information integration for intelligent systems (MFI), pp. 1–8. IEEE, Beijing (2014)
9. Barnard, S.T., Fischler, M.A.: Computational stereo. ACM Comput. Surv. (CSUR) **14**(4), 553–572 (1982)
10. Strydom, R., Thurrowgood, S., Denuelle, A., Srinivasan, M.V.: Uav guidance: a stereo-based technique for interception of stationary or moving targets. In: Conference towards autonomous robotic systems, pp. 258–269. Springer, Liverpool (2015)
11. Chen, X., Phang, S.K., Shan, M., Chen, B.M.: System integration of a vision-guided uav for autonomous landing on moving platform. In: IEEE international conference on control and automation (ICCA), 2016 12th, pp. 761–766. IEEE, Kathmandu (2016)
12. Benini, A., Rutherford, M.J., Valavanis, K.P.: Real-time, gpu-based pose estimation of a uav for autonomous takeoff and landing. In: IEEE international conference on robotics and automation (ICRA), 2016, pp. 3463–3470. IEEE, Stockholm (2016)
13. Cocchioni, F., Frontoni, E., Ippoliti, G., Longhi, S., Mancini, A., Zingaretti, P.: Visual based landing for an unmanned quadrotor. J. Intell. Robot. Syst. **84**(1-4), 511–528 (2016)
14. Benini, A., Mancini, A., Longhi, S.: An imu/uwb/vision-based extended kalman filter for mini-uav localization in indoor environment using 802.15. 4a wireless sensor network. J. Intell. Robot. Syst. **70**, 1–16 (2013)

15. Meguro, J.-I., Murata, T., Takiguchi, J.-I., Amano, Y., Hashizume, T.: Gps multipath mitigation for urban area using omnidirectional infrared camera. IEEE Trans. Intell. Transp. Syst. **10**(1), 22–30 (2009)
16. Wenzel, K.E., Masselli, A., Zell, A.: Automatic take off, tracking and landing of a miniature uav on a moving carrier vehicle. J. Intell. Robot. Syst. **61**(1), 221–238 (2011)
17. Hu, B., Lu, L., Mishra, S.: Fast, safe and precise landing of a quadrotor on an oscillating platform. In: American control conference (ACC), 2015, pp. 3836–3841. IEEE, Chicago (2015)
18. Dougherty, J., Lee, D., Lee, T.: Laser-based guidance of a quadrotor uav for precise landing on an inclined surface. In: American control conference (ACC), 2014, pp. 1210–1215. IEEE, Portland (2014)
19. Das, P.I.T.M., Swami, S., Conrad, J.M.: An algorithm for landing a quadrotor unmanned aerial vehicle on an oscillating surface. In: Southeastcon, 2012 proceedings of IEEE, pp. 1–4. IEEE, Orlando (2012)
20. Venugopalan, T., Taher, T., Barbastathis, G.: Autonomous landing of an unmanned aerial vehicle on an autonomous marine vehicle. In: Oceans, 2012, pp. 1–9. IEEE, Hampton Roads (2012)
21. Chaves, S.M., Wolcott, R.W., Eustice, R.M.: Neec research: toward gps-denied landing of unmanned aerial vehicles on ships at sea. Nav. Eng. J. **127**(1), 23–35 (2015)
22. Krajník, T., Vonásek, V., Fišer, D., Faigl, J.: Ar-drone as a platform for robotic research and education. In: International conference on research and education in robotics, pp. 172–186. Springer, Prague (2011)
23. Garratt, M., Pota, H., Lambert, A., Eckersley-Maslin, S., Farabet, C.: Visual tracking and lidar relative positioning for automated launch and recovery of an unmanned rotorcraft from ships at sea. Nav. Eng. J. **121**(2), 99–110 (2009)
24. Björck, Å.: Numerical methods for least squares problems. SIAM (1996)
25. Yakimenko, O.A., Kaminer, I.I., Lentz, W.J., Ghyzel, P.: Unmanned aircraft navigation for shipboard landing using infrared vision. IEEE Trans. Aerosp. Electron. Syst. **38**(4), 1181–1200 (2002)

Liyang Wang received the B.S. degree in department of electronic engineering from Beihang University in 2012, and received the M.S. degree in school of electronic, electrical and communication engineering from University of Chinese Academy of Sciences in 2015. His researches include quadrotor system design, sensor data fusion for flight state estimation, and autonomous control algorithms. He was involved in research project of UAV autonomous path planing and landing on vessel deck located at unknown area.

Xiaoli Bai has been an Assistant Professor in the department of Mechanical and Aerospace Engineering at Rutgers since July 2014. She obtained her PhD degree of Aerospace Engineering in 2010 from Texas A&M University. Her research theme is Advanced Computational Methods for Dynamics and Control of Aerospace Systems. Current research projects include the followings: (i) Advanced Orbit Prediction for Resident Space Objects (RSOs); (ii) Unmanned Aerial Vehicle (UAV) navigation and control; and (iii) Space robotic guidance, control, dynamics.